Verification of particle-in-cell simulations with Monte Carlo collisions

M. M. Turner

E-mail: miles.turner@dcu.ie
School of Physical Sciences and National Centre for Plasma Science and Technology, Dublin City University, Dublin 9, Ireland

Abstract. Widespread recent interest in techniques for demonstrating that computer simulation programs are correct ("verification") has been motivated by evidence that traditional development and testing procedures are disturbingly ineffective. Reproducing an exact solution of the relevant model equations is generally accepted as the strongest available verification procedure, but this technique depends on the availability of suitable exact solutions. In this paper we consider verification of a particle-in-cell simulation with Monte Carlo collisions. We know of no exact solutions that simultaneously exercise all of the functions of this code. However, we show here that there can be found in the literature a number of non-trivial exact solutions, each of which exercises a substantial subset of these functions, and which in combination exercise all of the functions of the code. That the code is able to reproduce these solutions is correctness evidence of a stronger kind than has hitherto been elucidated.

PACS numbers:

1. Introduction

The correctness of computer software is a matter of widespread concern, because there is as yet no approach to developing complex computer programs that guarantees to eliminate all errors. Consequently, systematic testing of computer programs is an important procedure. This concern has been sharpened in the case of scientific computations by the emergence of evidence that traditional testing procedures have not been effective in eliminating errors, even when the programs in question have been developed by teams of competent professionals, and used to inform economically consequential decisions ~~[10, 9]~~ [10, 12, 9, 5] . Evidence of this kind has motivated the question: What kind of testing will mount a strong defence against the presence of errors in a scientific code? The answer depends on what kind of problem we are seeking to solve. Many scientific calculations aim to compute a solution to some well-defined mathematical model, and in such cases, comparison with an exact solution to the same model supplies a powerful basis for testing a computer generated solution. Of course, this proposal opens the question of what should count as "agreement" between an exact solution and a solution computed using finite precision arithmetic. However, if the algorithm in use is well understood, we may be able to predict how the error in the computed solution should change as some numerical parameter is varied. For instance, if the algorithm depends on a finite time step, Δt, and we know that the associated error is $O(\Delta t^2)$, then we expect that the distance between the exact solution and the computed solution (expressed using some suitable metric) will be proportional to Δt^2. Hence we can ask for a demonstration that the error indeed scales in this fashion. This will be the case if and only if the calculation actually converges to the correct solution at the expected rate. Such a test is strenuous, and will disclose many subtle implementation errors [19]. Some claim (although controversially) that a test of this kind should be accepted as a correctness proof [19]. Whether we believe this or not, a test using this procedure is the most compelling evidence of correctness that can presently be put forward. An obvious difficulty is that often we do not know a relevant exact solution. For an important category of problems, this obstacle has been overcome by the Method of Manufactured Solutions [19, 7]. This technique follows from the insight that the exact solution that we test against need not be physical. The method is to choose a solution almost arbitrarily, and insert a special source term into the model equations that will force the chosen solution to become an exact solution. Hence, provided we can add this source term, we can establish convergence to an exact solution for almost any system of equations. This approach works for most, if not all, problems involving systems of partial differential equations, and is widely used in the field of computational fluid dynamics, for instance. However, there are difficulties in using this approach with Monte Carlo codes, such as particle-in-cell simulations. Implementing the necessary source terms is likely to be challenging, for example, and this is not the only difficulty. So far, no one has demonstrated that the Method of Manufactured Solutions can be used in this context. So we may usefully seek other ways of improving confidence in the

correctness of kinetic codes.

The present author and a group of collaborators recently developed a benchmark [21] applicable to a class of codes implementing the particle-in-cell with Monte Carlo collisions procedure [11, 2, 3]. We constructed four benchmark cases related to capacitive discharges in helium. The physical and numerical parameters were fully specified for each case. We showed that five independently written implementations of the procedure produced results that were statistically indistinguishable from each other. We have since heard of independent reproductions of the benchmarks by other researchers. This might appear to be compelling evidence that the implementations involved were fault-free. However, this view has met with objections, essentially on two grounds. The first is that even if the algorithm has been implemented as designed in all cases, correct physics is not guaranteed to follow. In other words, there may be conceptual error. Clearly, multiple independent implementation is no protection in this case. A second objection is that the independent implementations may in fact be simultaneously faulty. The probability of such an eventuality has been studied in the context of decision making systems, and indeed independent implementations tend not fail in a statistically independent fashion [13]. This occurs for reasons that are not clearly understood: Perhaps certain algorithmic units are psychologically harder to implement correctly, and faults cluster in these areas. If the frequency of faults observed in such studies is any guide in the present context, five or more simultaneous failures would be a far fetched possibility. But we have no evidence directly relevant to simultaneous failures in scientific codes, so a strong rebuttal of this objection on probabilistic grounds cannot be made at the moment. These arguments suggest that further evidence that the benchmark calculations are correct could usefully be advanced, and that this evidence should preferably involve demonstrating convergence to exact solutions in the manner discussed above. Such a procedure would meet both of these objections.

For reasons already discussed, the Method of Manufactured Solutions cannot yet be used with Monte Carlo codes. We therefore sought other avenues. Two complementary groups of tests will be discussed. The first group consists of unit tests. These tests involve a single component of the simulation in isolation, such as the integrator for particle motion, the solution of the field equation or random number generators. These are straightforward demonstrations that a (usually quite elementary) outcome is properly reproduced by the component in question. Most of these tests antedate the development of the benchmarks, but for reasons of space they were not discussed in the benchmark paper. In view of the sceptical objections outlined above, it appears desirable to document these tests, albeit briefly. Unit tests go only so far towards demonstrating the correctness of the whole implementation, which combines the units in complicated ways. The second group of tests consists of demonstrations of convergence towards non-trivial solutions of the underlying kinetic equations that the particle-in-cell procedure is intended to solve. These solutions are drawn from the applied mathematics literature. None of these solutions combines all the units involved in the benchmark calculations, but each combines several such units. This category of solutions thus lies between

the unit tests and the full scale problems that the codes in question are designed to solve, such as the benchmark cases, and we may therefore refer to them as "mezzanine" problems.

The remainder of this paper us structured in the following way. Section 2 discusses the unit tests, but without entering into full detail. Section 3 describes three mezzanine tests in some detail. Section 4 discusses some issues arising from the application of the tests, and section 5 presents conclusions.

2. Unit tests

A unit test is designed to verify a component of a larger simulation program in isolation. For components that have a numerical function, such a test is likely to involve a demonstration of convergence to an exact solution, for reasons discussed in section 1. For instance, the algorithms used for integrating the particle equations of motion are designed to be second order accurate with respect to the time step, Δt, and therefore an appropriate verification test will involve demonstrating convergence to an exact solution of the relevant equations. Similarly, the solution of the field equation is designed to be second order accurate with respect to the cell size, Δx, and so verification of the field equation solution proceeds analogously. Several test cases are needed to fully exercise the field solver, because several different boundary conditions are permitted, and each requires seperate verification. Less complicated tests include verification that the collision operators conserve energy and momentum as they are supposed to. The random number generator used in these calculations is the Mersenne twister [15]. We have assumed that this generator is not in need of verification by us. We have, however, verified that the various derived random number distributions that are employed, for example in collision handling, do produce the expected results. An appropriate tool for this purpose is the X^2 test [1, sec. 26.4] (also discussed below in section 3.3).

We will not discuss all of these unit tests in detail. We will, however, present one example. This is a verification test for integration of particle equations of motion in electric and magnetic fields. There is an exact solution for the case where the magnetic field is stationary and the electric field varies harmonically in time [16, chap. 2]. The code under investigation in the present study implements the Boris method of integrating the particle equations of motion[2], which is designed to be second order accurate with respect to the time step Δt. Since the electric and magnetic fields are prescribed (and uniform in space), this is the only relevant numerical parameter. The distance between the numerical solution and the exact solution can be measured by the L^2 relative error norm:

$$L^2 = \sqrt{\frac{\sum_i \left[v_{\text{numerical}}(t_i) - v_{\text{exact}}(t_i)\right]^2}{\sum_i v_{\text{exact}}(t_i)^2}} \tag{1}$$

where $t_i = (i - \frac{1}{2})\Delta t$, because the Boris method is a leap-frog algorithm that defines velocities at half-integral time steps. If the code correctly implements the Boris

integration scheme, in the sense that there is convergence to the exact solution at the expected rate, we expect

$$L^2 \propto \Delta t^2. \tag{2}$$

Relatively subtle errors cause this test to fail. For example, overlooking the need to properly time align the position and velocity at the start of the integration will lead to $L^2 \propto \Delta t$. Consequently, a strong verification test for this time integration scheme is to calculate solutions for a sequence of different step sizes, and then demonstrate that the L^2 error norm scales as predicted by equation 2. Figure 1 shows such a sequence of solutions, and figure 2 shows that the L^2 error norm indeed varies in the expected way. For the particular case shown here, the cyclotron frequency $\omega_c = 10^9$ Hz and the magnetic field is at an angle $\phi = \pi/4$ relative to the x axis. The electric field is parallel to the x axis and oscillates with a frequency $\omega = \omega_c/4$.

3. Mezzanine tests

3.1. Criticality

Neutron transport is an important topic in the context of nuclear engineering, and has consequently been the object of intensive study. The mean free path of a neutron in fissile materials such as uranium or plutonium is a few centimeters, and the radius of a sphere containing a critical mass of these metals is of the same order of magnitude. Consequently, an accurate determination of the critical mass requires the solution of a kinetic transport problem. This consideration has motivated intensive mathematical work, which has discovered several exact solutions of the kinetic equation for neutron transport [4, 20]. These solutions refer to simplified situations, but this does not reduce their utility as verification tests for kinetic computer simulation codes. The mathematics involved is elaborate [4], and will not be discussed here, but the problem statements are easily expressed. Three neutron scatttering processes are considered, namely elastic scattering, capture, and fission. Cross sections are specified for each of these processes, together with the average number of neutrons produced by each fission event. The neutrons are supposed to be monoenergetic, and to sustain this assumption, scattering is assumed to involve collision partners of infinite mass. When this combination of processes occurs in an object of finite size, there is a competition between neutron multiplication in fission events, and loss by transport and by capture. In a simple geometry (such as a slab, a cylinder, or a sphere) there is a single characteristic dimension, and at a certain value of this dimension, transport losses and multiplication by fission are exactly in balance. This is known as the critical condition. Mathematically, this is an eigenvalue problem, for which, as already mentioned, exact solutions are known, and have been collated for the purpose of verifying neutron transport codes[20]. However, these solutions are also useful for verifying important parts of the collision handling apparatus in a collisional particle code, including proper implementation of isotropic scattering, injection of particles following inelastic collisions, partition of energy

and momentum between particles after collisions, collision selection procedures, and calculation of intervals between collisions. Of course, correct implementation of particle transport is also essential.

In this work we consider the criticality problem for plutonium in a slab geometry. We assume that the boundaries are fully absorbing, and that neutrons are scattered isotropically in the laboratory frame. The nuclear chemistry model involves three reactions:

$$n + \mathrm{Pu} \to n + \mathrm{Pu} \tag{3}$$

$$n + \mathrm{Pu} \to \mathrm{Pu} \tag{4}$$

$$n + \mathrm{Pu} \to 3.24n + \text{fission products}, \tag{5}$$

where we implemented the last reaction as a linear combination of processes emitting three and four neutrons, weighted to achieve the indicated mean neutron yield. The problem statement prescribes recriprocal mean free paths for each of these processes, which are ~~22.5216~~22.52 m^{-1}, ~~8.16~~8.160 m^{-1} and ~~1.9584~~1.960 m^{-1}, giving a neutron mean free path of approximately 3 cm. Once these nuclear processes have been specified, the criticality problem in any given geometry is to find the characteristic (critical) system size at which the neutron density becomes stationary in time, due to a balance between volume production and loss by transport across the boundaries. Of course, for characteristic sizes less than the critical value, the neutron density falls, and for sizes above the critical value, it increases. In the case of the slab considered here, this critical thickness is found analytically to be ~~3.707444~~3.707 cm. Since this is only slighter larger than the mean free path, the kinetic character of the problem is clear. In a computer simulation, we do not expect to find a precisely stationary neutron density, but rather for sufficiently long times $n \sim \exp(-t/\tau)$, where by convention we assume that $\tau > 0$ when the neutron density is decaying. Even if the neutron density is never stationary, we can show that the rate of change of the neutron density converges to zero as the numerical parameters are suitably refined. A convenient reference time for these calculations is the decay time of the neutron density with respect to transport processes only, which is $\tau_0 \approx 3 \times 10^{-5}$ s.

For this problem, the numerical procedure for integrating the particle equations of motion is exact (or really, limited by round-off error). Consequently, the only inexact step in the calculation is the treatment of collisions, which in the usual way[3] are assumed to be governed by a collision probability per time step given by

$$P = 1 - \exp(-\nu \Delta t), \tag{6}$$

for particle collision frequency ν and time step Δt. This procedure prohibits the occurence of more than one collision per time step, and consequently is first order accurate in time. Hence we expect to find an error that is proportional to the time step. Since the effect of this error is to reduce the collision frequency (and hence the fission rate), we expect to find that the neutron density decays ($\tau > 0$) when the simulation parameters correspond to the exact criticality condition, but that extrapolation of the

decay rate to zero time step should yield a value consistent with the criticality condition. This demonstration is complicated by statistical effects in the simulation, of course.

The criticality condition holds for all neutron speeds, so any value can be chosen. We selected a speed such that $\frac{1}{2}mv^2 = \frac{3}{2}k_BT$ for $T = 300$ K. This choice affects the absolute value of the collision frequency and the time step, but is otherwise of no significance. The cell size was chosen such that the normalized neutron speed $v\,\Delta t/\Delta x \approx 1$ was independent of the time step. An example of the quasi-stationary neutron density observed in the present calculations is shown in figure 3. Figure 4 shows the neutron decay rate as a function of the simulation time step. In this figure, the error bars represent the standard deviation of an ensemble of statistically independent simulations executed using the same numerical parameters. The solid line is a least squares fit taking account of the error bars, which gives an intercept on the vertical axis of $1/\tau = (4.2 \pm 4.7) \times 10^{-5}/\tau_0$, so that $\tau \approx 25000\tau_0$, with an error bar consistent with $\tau = \infty$.

3.2. Plane Diode

This test problem describes space-charge limited flow through a plane diode with two grounded walls, each assumed to be an infinite plane. A prescribed flux of electrons with a Maxwellian velocity distribution is emitted from one boundary. When an emitted electron recrosses either boundary, reabsorption is assumed to occur. The presence of the electrons in the gap produces a negative space-charge. Therefore, the potential between the walls is also negative, and this negative potential limits the flow of electron across the gap. Consequently, the current that flows through the diode depends on the size of the gap, and the characteristics of the emitted flux. Since the flow is assumed to be collisionless, the governing equations are the Vlasov-Poisson system. An exact solution has been discovered for this problem, which expresses the current density passing through the diode in terms of basic physical parameters of the system [8, 6, 18]. The speed distribution for particles of mass m and charge q can be written as

$$f(x,v) = \frac{n_0}{v_t\sqrt{\pi}} \exp\left[-\left(v^2v_t^2 + \frac{2q\phi(x)}{mv_t^2}\right)\right], \tag{7}$$

where n_0 is the electron density adjacent to the emitting boundary, $v_t = \sqrt{2k_BT_0/m}$, and T_0 is the temperature of injected particles. The current density is found by taking the first moment of the distribution function, and is

$$J_0 = \frac{qn_0v_t}{2\sqrt{\pi}} \exp\left(-\frac{2q\phi_{\rm m}}{mv_t^2}\right), \tag{8}$$

where $\phi_{\rm m}$ is the maximum value of the potential. The current density is thus controlled by the dimensionless parameter

$$\Phi_{\rm m} = \frac{2q\phi_{\rm m}}{mv_t^2}, \tag{9}$$

which is itself a function of the normalized system length (in SI units):

$$\Lambda = \frac{L}{\sqrt{\epsilon_0 k_B T_0 / e^2 n_0}} = \frac{L}{\lambda_D}. \tag{10}$$

The relationship between Φ_m and Λ involves a quadrature [18], the result of which is shown in figure 7.

The verification problem is to show that the simulation result converges to the exact solution as the numerical parameters are suitably refined. This entails that a specific example be chosen, and (following Radtke et al. [18]) we choose $\Lambda = 20$, which gives ~~$\Phi_m = 2.39792162362$. For $q = 1.6022 \times 10^{-19}$~~ $\Phi_m = 2.398$. For $q = 1.602 \times 10^{-19}$ C, $m = 9.109 \times 10^{-31}$ kg, $T_0 = 10$ eV and $n_0 = 10^{16}$ m^{-3}, this gives ~~$J_0 = 77.0608433$~~ $J_0 = 77.06$ A m~~$^{-2}$~~$^{-2}$. Three numerical parameters are concerned in this problem, namely the time step, cell size, and number of particles per cell. The numerical error is expected to be quadratic with respect to refinement of each of these parameters, and in this work, to avoid complex procedures, we refine all three parameters together. The base conditions are $\omega_p \Delta t = 0.8$, Δx corresponding to thirty two cells, and four particles per cell, where these parameters are defined relative to the reference density n_0. Under the base conditions, the simulation is executed for 2^{18} steps, with a correspondingly larger number for refined cases. This simulation time is more than a thousand times longer than the transit time of a thermal electron. At each refinement, the number of cells and the number of particles per cell is doubled, while the time step is halved. For each such case, we compute ΔJ_0, defined as the difference between the current densities specified by the exact solution and observed in the simulation. The current density is observed only in the second half of the simulation period, so that transient effects are suppressed. The outcome of the process of refinement is shown in figure 8, where the error bars derive from the standard error of the mean of the ensemble of current density samples. A least squares fitting procedure is used to extrapolate the refinement to zero time step. The intercept on the vertical axis given by this method is

$$\Delta J / J_0 = (-0.8 \pm 2.6) \times 10^{-4}, \tag{11}$$

which clearly is consistent with the expected outcome of zero. Thus we conclude that any implementation errors have effects smaller than the statistical resolution of the calculation, which itself corresponds to a considerably smaller uncertainty than is usually expected from a particle-in-cell simulation in practical usuage.

3.3. Ion Swarm

In this example we consider a group of particles (ions) that are accelerated by a time varying field such that

$$E(t) = \tilde{E} \cos \omega t = \tilde{E} \cos \tau. \tag{12}$$

The particles have charge q and mass m. They collide elastically at frequency ν with cold collision partners of the same mass, and are backward scattered in the centre of mass reference frame. This apparently trivial situation is of interest because the particle

velocity distribution functions that occur are surprisingly intricate and can be described by an exact solution of the associated kinetic equation. Reproducing these distribution functions is consequently a suitable verification test problem. The exact solution of the kinetic equation under these conditions[14] is:

$$f_\gamma(u,\tau) = \begin{cases} 0 & \text{if } |w_0| > 1 \\ \dfrac{\gamma}{1-\exp(2\pi\gamma)}\left[\dfrac{\exp(-\gamma\Delta\tau_1)+\exp(-\gamma\Delta\tau_2)}{\sqrt{1-w_0^2}}\right] & \text{otherwise,} \end{cases} \tag{13}$$

where $\gamma = \nu/\omega$ and $w_0 = u - \sin\tau$. In the first quadrant ($0 \le \tau < \pi/2$), $\Delta\tau_1$ and $\Delta\tau_2$ are given by

$$\Delta\tau_1(u,\tau) = \tau + \pi - \arcsin w_0 \tag{14}$$

$$\Delta\tau_2(u,\tau) = \begin{cases} \tau + 2\pi + \arcsin w_0 & \text{if } u \le 0 \\ \tau + \arcsin w_0 & \text{otherwise.} \end{cases} \tag{15}$$

Straightforward symmetry arguments extend these results to other values of τ. Equation 13 is an exact, normalized, solution of the kinetic equation. This solution has two unusual features: It is in general discontinuous across the axis $u = 0$, and it is singular at the edges of the region where it is non-zero, that is, where $w_0 \to \pm 1$.

The verification problem is to demonstrate that the simulation program can reproduce the distribution function given by equation 13, for some particular values of τ and γ. As we can only sample this distribution in the simulation from a finite number of particles, this will amount to testing the hypothesis that the ensemble of particles we observe have velocities drawn from the distribution function given by equation 13. A suitable tool in this context is the X^2 test. For a set of k observations given by O_i and a corresponding set of expected values given by E_i:

$$X^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}. \tag{16}$$

This value fluctuates when there are statistical variations in the observed values, with the effect that if the observations are consistent with the expected values, then the distribution of X^2 values is given by

$$f(x,k) = \begin{cases} x\dfrac{x^{(k/2-1)}\exp(-\frac{x}{2})}{2^{k/2}\Gamma\left(\frac{k}{2}\right)}, & x > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

where

$$\int_0^\infty f(x,k)\,dx = 1. \tag{18}$$

For a single trial, one must judge whether the resulting value of X^2 is improbable when evaluated against the distribution of equation 17. When computational limitations permit, however, a less ambiguous test is to compute an ensemble of values of X^2 and investigate whether they are consistent with the distribution given by equation 17. This

is the procedure we follow here. A convenient simplification is achieved by dividing the expected distribution function into equiprobable intervals, such that E_i is the same for each interval. In the present case, this is facilitated by the existence of an exact cumulative probability distribution, *i.e.* there exists an exact integral of equation 13.

Figure 9 compares a sample of the velocity distribution obtained from the simulation with the exact result given by equation 13. Clearly, the agreement appears good, but not exact. Of course, we do not expect exact agreement, even if the simulation result is correct, because of statistical fluctuations. The X^2 test provides a means to determine whether the difference between the two distributions is of a purely statistical character, or not. In the latter case, we would need to consider the origin of the difference, which could, for instance, be due to incorrect code, incorrect data, or to insufficient convergence with respect to some numerical parameter. For the present calculation, figure 10 shows that for an ensemble of simulated velocity distribution functions, the distribution of X^2 values indeed follows the expected distribution. This means that any influences of faulty coding, incorrect data or finite numerical parameters are small compared to statistical effects. As an indication of the sensitivity of this test, if one compares the simulated distribution with the exact distribution evaluated at a different phase, a shift of $\Delta\Phi = 0.02\pi$ produces an easily appreciable change in the distribution of X^2 values.

4. Discussion

In the introductory remarks, we noted that the practice of comparing independent codes meets with objections, if presented as a verification exercise. A method that addresses these objections is a comparison with an exact analytic solution, where the comparison is to be made by demonstrating convergence towards the expected solution at the rate predicted from a theoretical understanding of the behaviour of the algorithm employed. Ideally, the exact solution should exercise all the functions of the code under test. Alas, we know of no solutions that meet this condition. However, we have seen that the applied mathematics literature contains a number of exact solutions for problems that exercise substantial subsets of the functions of a collisional particle code. These are intermediate or "mezzanine" solutions. Testing against these solutions is stronger than carrying out unit tests directed toward specific functions, because the functions are combined in non-trivial ways, *i.e.* more of the code is being tested. In combination, the mezzanine test cases discussed here cover the principal functions involved in solving the benchmark problems presented in an earlier paper [21]. Table 1 summarizes this coverage of the test problems. Probably the most powerful of these mezzanine tests is the criticality problem, because this test exercises the important case of isotropic scattering together with the handling of inelastic processes involving the addition and removal of particles. Uniquely among the unit tests and the mezzanine tests considered here, the criticality problem also involves the computation of a distribution function that is three dimensional in velocity space [4]. A further attractive feature of this test case is that the straightforward character of the problem statement conceals the very considerable

mathematical sophistication involved in the construction of the solution [4]. That such a useful test problem was found in literature outside the sphere of obvious relevance suggests that the search for further useful test problems should be conducted rather broadly. Very likely, problems of equal or better value exist in the applied mathematics literature.

In the present work we have taken a rather unsophisticated approach to demonstrating consistency with the exact solutions, in the sense that our test problems are constructed so that the error is dominated by either effects associated with numerical parameters (in most of the unit tests, the criticality problem, and the plane diode problem), or by statistical effects (in the ion velocity distribution problem, and in tests of random number generators). These methods appear satisfactory for present purposes, but a more general method would be desirable. How this should be done is presently a research problem. One approach has recently been described by Radtke et al. [18], and applied to the plane diode problem discussed above.

5. Conclusions

The aim of the present paper is to increase confidence in the benchmark solutions presented earlier[21], by showing that one of the codes involved in the original work passes a set of verification tests involving convergence to exact solutions of various test problems. This procedure is generally accepted to be a strong test of code correctness[17]. Two kinds of problems have been considered. Unit tests involve only single component of the code, and thus exercise the code in a relatively limited fashion. Mezzanine tests are more complex problems, involving several components of the code operating correctly together, and thus present a greater challenge. These tests in combination have revealed no faults with any implications for the validity of the benchmark calculations[21]. This result, at the least, reduces the scope for the objections mentioned in the introduction, and consequently increases confidence in the validity of the benchmarks. Of course, the discovery of more comprehensive exact solutions, or the resolution of the difficulties presently attending the use of the Method of Manufactured Solutions, remain as desirable objectives for future work.

Acknowledgments

The author's attention was drawn to the usefulness of the plane diode problem by Keith Cartwright of Sandia National Laboratories, who also derived the form of the solution quoted in the text above.

References

[1] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*. Dover Publications,

New York, 0009-revised edition edition, June 1965. ISBN 978-0-486-61272-0.

[2] Charles K. Birdsall and A. Bruce Langdon. *Plasma physics via computer simulation*. Adam Hilger, 1991. ISBN 978-0-7503-0117-6.

[3] C.K. Birdsall. Particle-in-cell charged-particle simulations, plus Monte Carlo collisions with neutral atoms, PIC-MCC. *IEEE Transactions on Plasma Science*, 19(2):65–85, April 1991. ISSN 0093-3813. doi: 10.1109/27.106800.

[4] K. M Case. Elementary solutions of the transport equation and their applications. *Annals of Physics*, 9(1):1–23, January 1960. ISSN 0003-4916. doi: 10.1016/0003-4916(60)90060-9.

[5] Geoffrey Chang, Christopher B. Roth, Christopher L. Reyes, Owen Pornillos, Yen-Ju Chen, and Andy P. Chen. Retraction. *Science*, 314(5807):1875b–1875b, December 2006. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.314.5807.1875b.

[6] Pierre Degond. The Child Langmuir Law in the Kinetic Theory of Charged Particles. In *Advances in kinetic theory and computing*, volume 22 of *Series on Advances in Mathematics for Applied Sciences*, pages 3–44. World Scientific, 1994.

[7] B. D. Dudson, J. Madsen, J. Omotani, P. Hill, L. Easy, and M. Løiten. Verification of BOUT++ by the method of manufactured solutions. *Physics of Plasmas (1994-present)*, 23(6):062303, June 2016. ISSN 1070-664X, 1089-7674. doi: 10.1063/1.4953429.

[8] Claude Greengard and P.-A. Raviart. A boundary-value problem for the stationary vlasov-poisson equations: The plane diode. *Comm. Pure Appl. Math.*, 43(4):473–507, June 1990. ISSN 1097-0312. doi: 10.1002/cpa.3160430404.

[9] L. Hatton. The T experiments: errors in scientific software. *IEEE Computational Science and Engineering*, 4(2):27–38, April 1997. ISSN 1070-9924. doi: 10.1109/99.609829.

[10] L. Hatton and A. Roberts. How accurate is scientific software? *IEEE Transactions on Software Engineering*, 20(10):785–797, October 1994. ISSN 0098-5589. doi: 10.1109/32.328993.

[11] Roger W. Hockney and James W. Eastwood. *Computer Simulation Using Particles*. McGraw-Hill, 1981. ISBN 978-0-07-029108-9.

[12] Bernt Jakobsen. The Sleipner accident and its causes. *Engineering Failure Analysis*, 1(3):193–199, October 1994. ISSN 1350-6307. doi: 10.1016/1350-6307(94)90018-3.

[13] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1):96–109, January 1986. ISSN 0098-5589. doi: 10.1109/TSE.1986.6312924.

[14] Kailash Kumar. Swarms in Periodically Time Dependent Electric Fields. *Aust. J. Phys.*, 48(3):365–376, January 1995.

[15] Makoto Matsumoto and Takuji Nishimura. Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator. *ACM Trans. Model.*

	Criticality	Plane Diode	Ion Swarm	Benchmark
Time integration	✓	✓	✓	✓
Field solution		✓		✓
Collision handling	✓		✓	✓
Isotropic scattering	✓			✓
Backward scattering			✓	✓
Elastic collisions	✓		✓	✓
Inelastic collisions	✓			✓

Table 1. Major functions of a collisional particle-in-cell simulation, as exercised in the three test cases discussed in the present paper, and the benchmark calculations of Turner et al. [21].

Comput. Simul., 8(1):3–30, January 1998. ISSN 1049-3301. doi: 10.1145/272991.272995.

[16] Michel Moisan and Jacques Pelletier. *Physics of Collisional Plasmas: Introduction to High-Frequency Discharges*. Springer Science & Business Media, June 2012. ISBN 978-94-007-4558-2.

[17] William L. Oberkampf and Christopher J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, October 2010. ISBN 978-1-139-49176-1.

[18] Gregg A. Radtke, Keith L. Cartwright, and Lawrence C. Musson. Stochastic Richardson Extrapolation Based Numerical Error Estimation for Kinetic Plasma Simulations. Sandia Report SAND2015-8620, Sandia National Laboratories, Albuquerque, New Mexico 87185, U. S. A., October 2015.

[19] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions. *J. Fluids Eng*, 124(1):4–10, November 2001. ISSN 0098-2202. doi: 10.1115/1.1436090.

[20] Avneet Sood, R. Arthur Forster, and D. Kent Parsons. Analytical benchmark test set for criticality code verification. *Progress in Nuclear Energy*, 42(1):55–106, 2003. ISSN 0149-1970. doi: 10.1016/S0149-1970(02)00098-7.

[21] M. M. Turner, A. Derzsi, Z. Donkó, D. Eremin, S. J. Kelly, T. Lafleur, and T. Mussenbrock. Simulation benchmarks for low-pressure plasmas: Capacitive discharges. *Physics of Plasmas (1994-present)*, 20(1):013507, January 2013. ISSN 1070-664X, 1089-7674. doi: 10.1063/1.4775084.

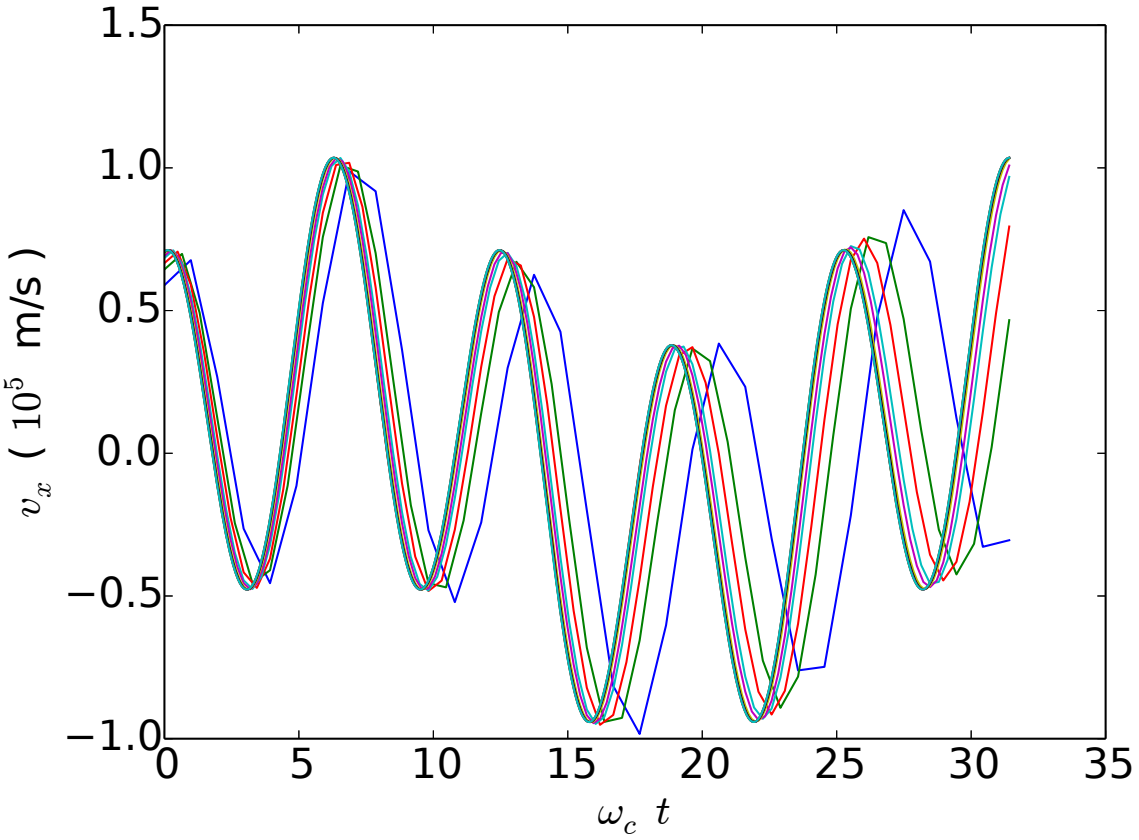

Figure 1. A sequence of numerical solutions for the motion of a charged particle in a static magnetic field and a harmonic electric field. The integration time step is halved at each step in the sequence, and the solution consequently converges to the exact result, as shown in fig. 2.

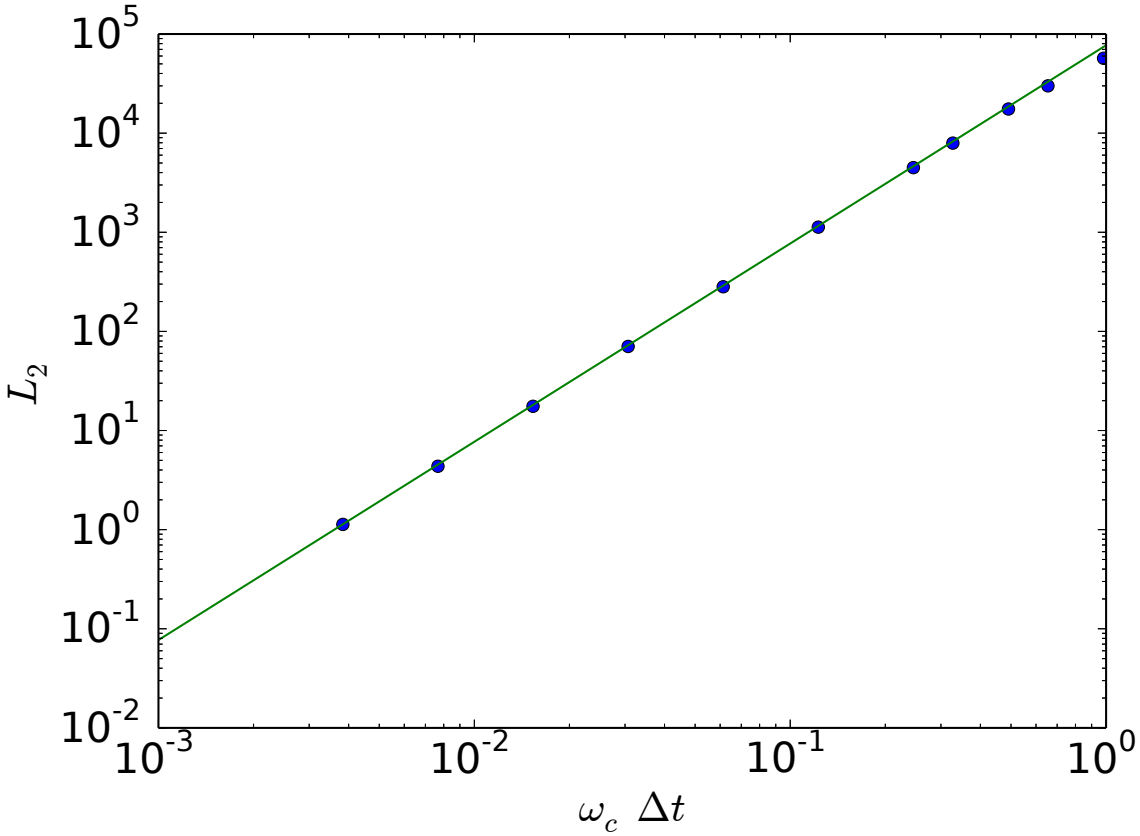

Figure 2. The L^2 error norm for the sequence of solutions shown in figure 1, calculated relative to the exact solution. The points correspond to the sequence of L^2 values, while the line shows the expected scaling $\propto \Delta t^2$. At $\omega_c\,\Delta t = 1$, the data point is below the curve because the regime of asymptotic convergence has not been reached.

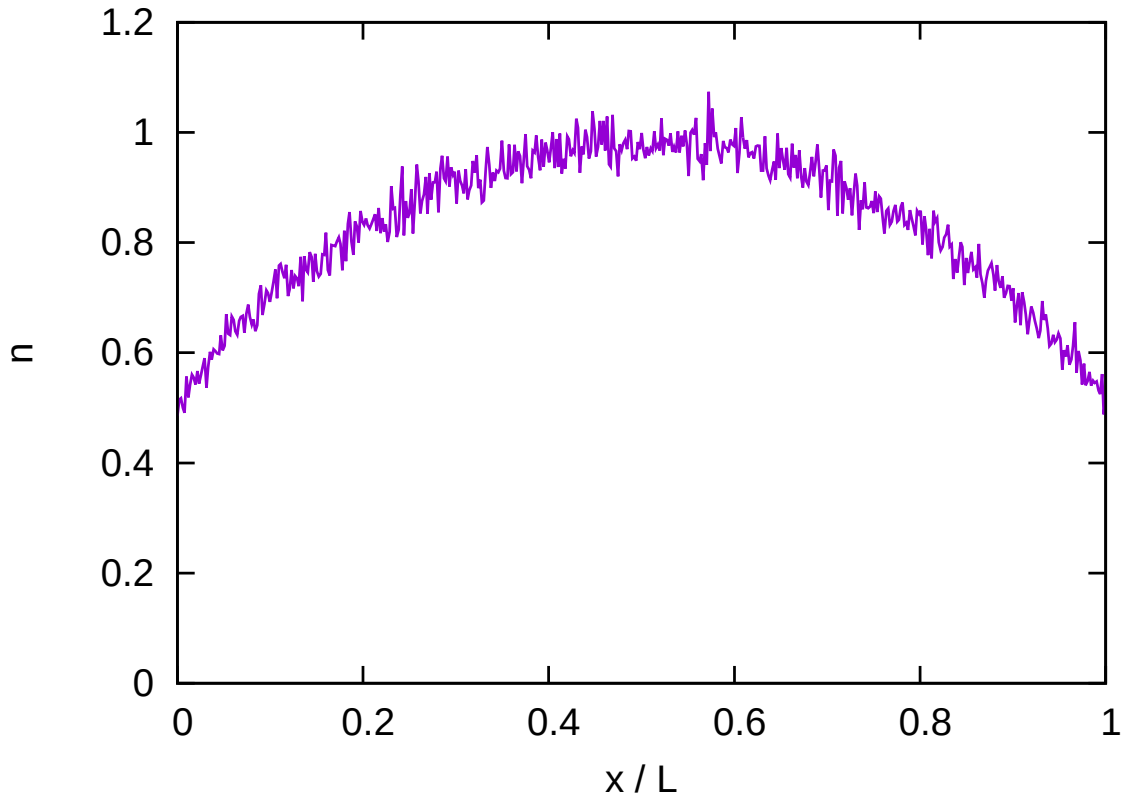

Figure 3. An example of the quasi-stationary neutron density obtained by simulating the critical condition discussed in section 3.1.

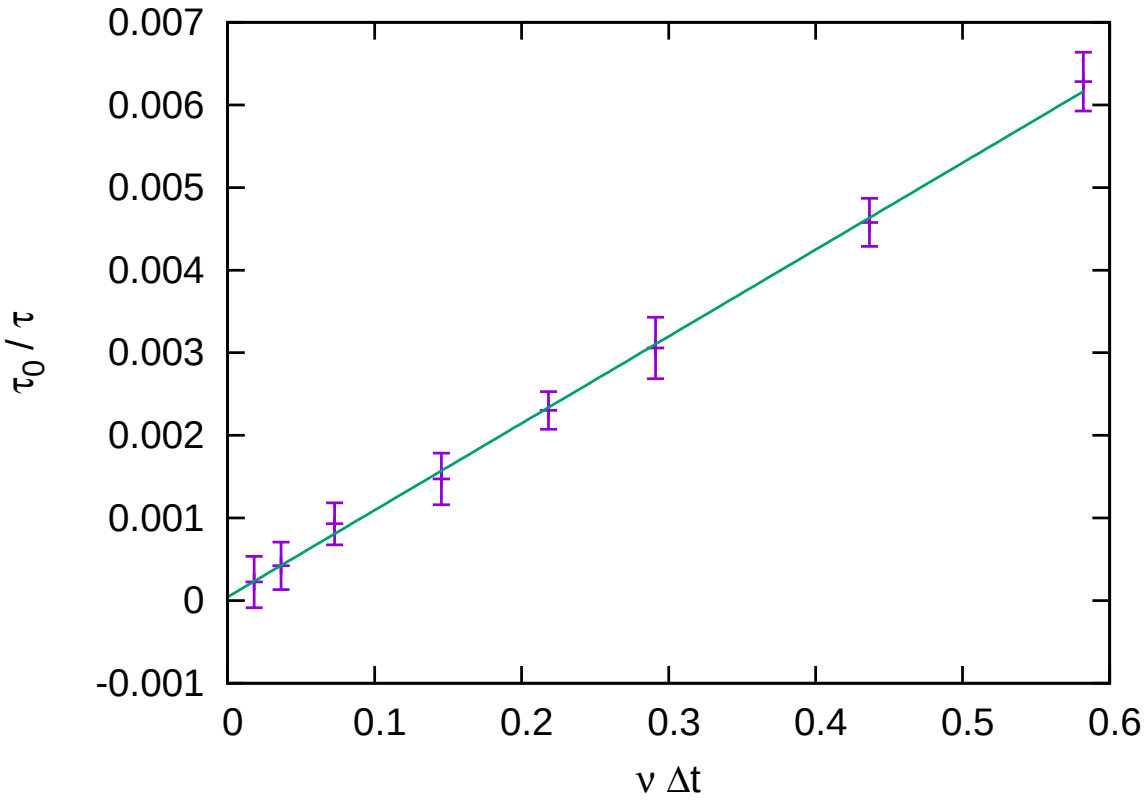

Figure 4. The decay constant observed in the simulation under conditions corresponding to the analytically determined critical condition (points with error bars). Expected behaviour is

$$\lim_{\nu\Delta t\to 0} \tau_0/\tau = 0$$

with a linear rate of convergence. The solid curve is a least squares fit to the simulation data, from which the limiting value is estimated to be $(4.2 \pm 4.7) \times 10^{-5}$.

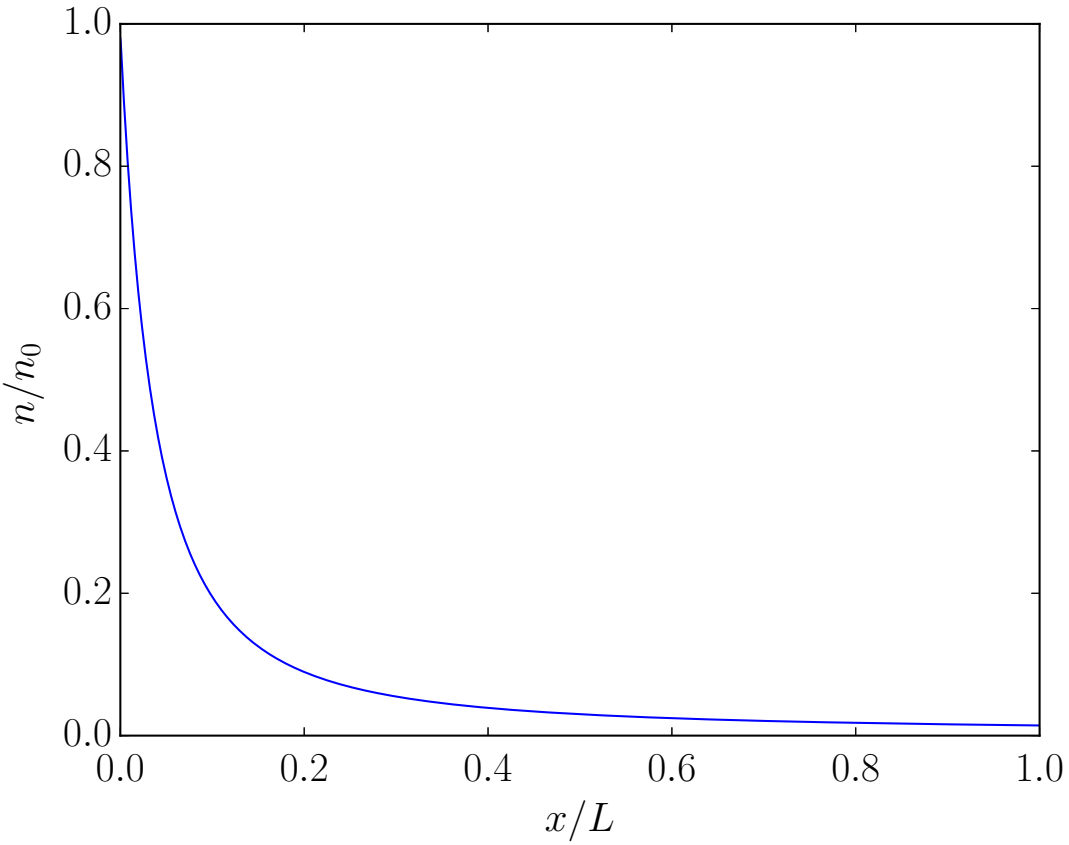

Figure 5. The electron density in the planar diode discussed in the text. Both walls are grounded, and electrons with a Maxwellian velocity distribution are injected from the left boundary.

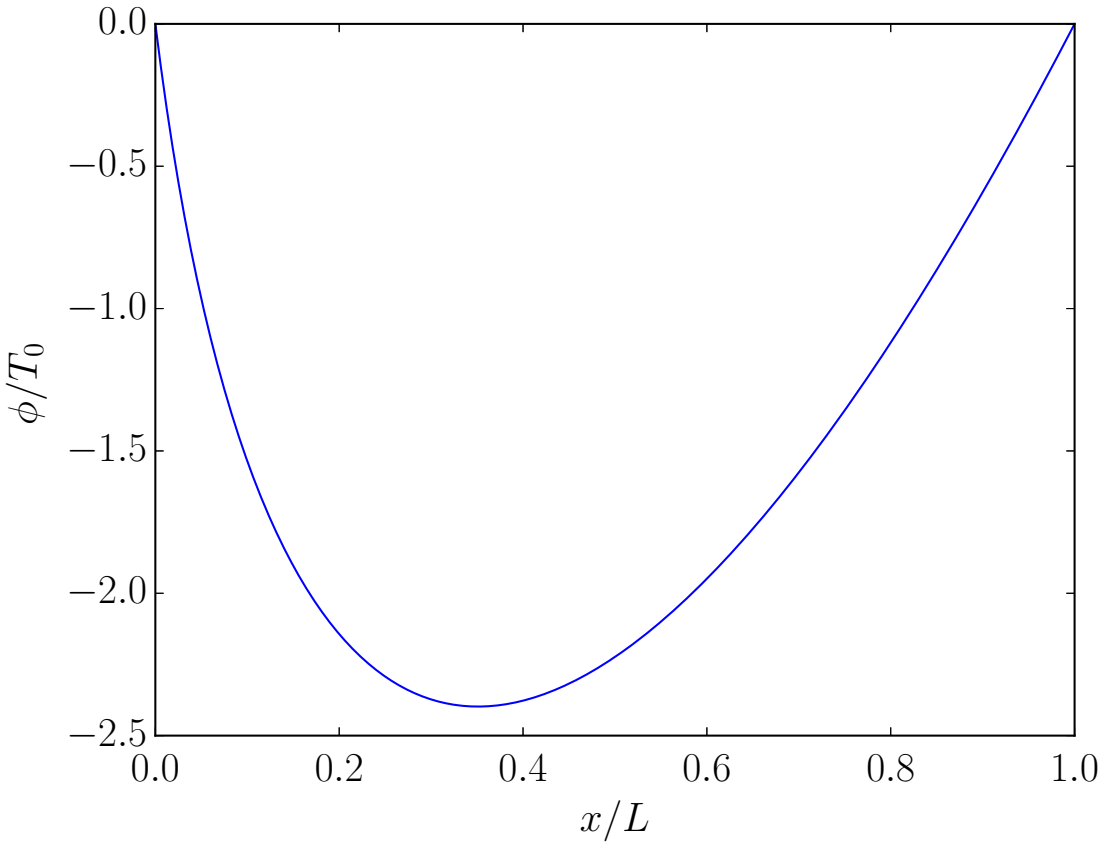

Figure 6. The potential corresponding to the electron density distribution shown in figure 5.

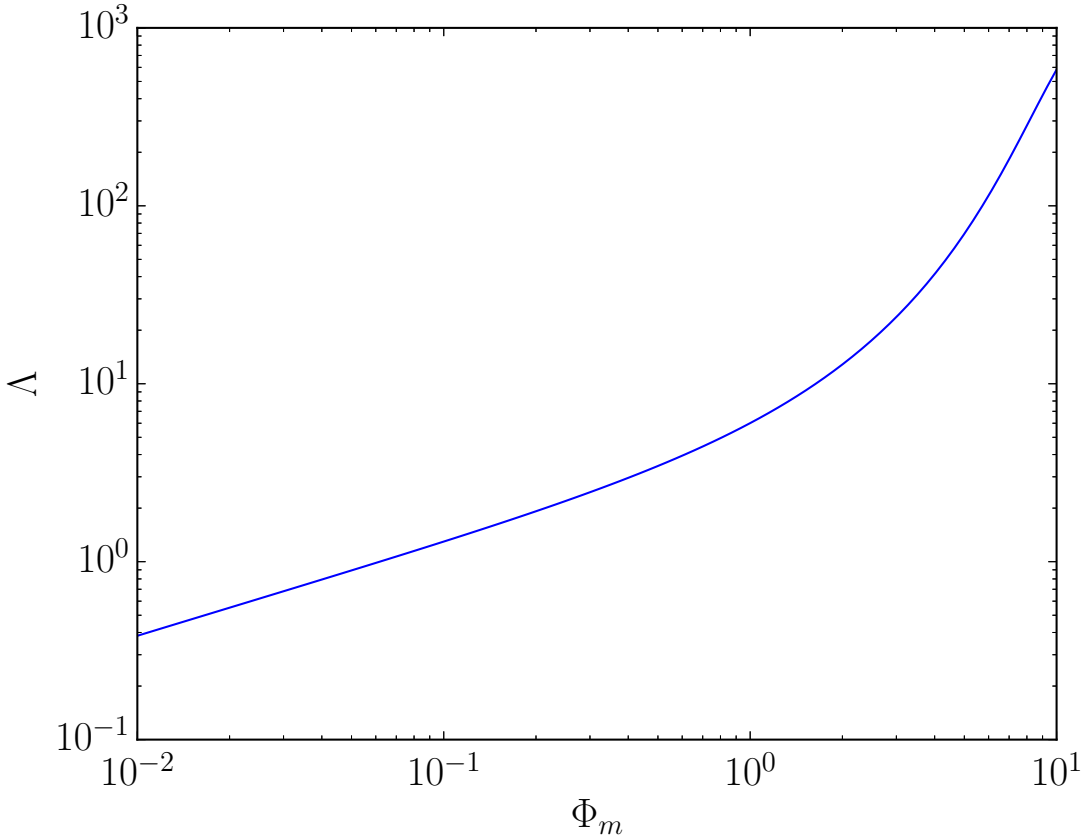

Figure 7. Relationship between the normalized maximum potential $\Phi_{\mathrm{m}} = e\phi/K_BT$ and the normalized system length $\Lambda = L/\lambda_D$ for the plane diode problem.

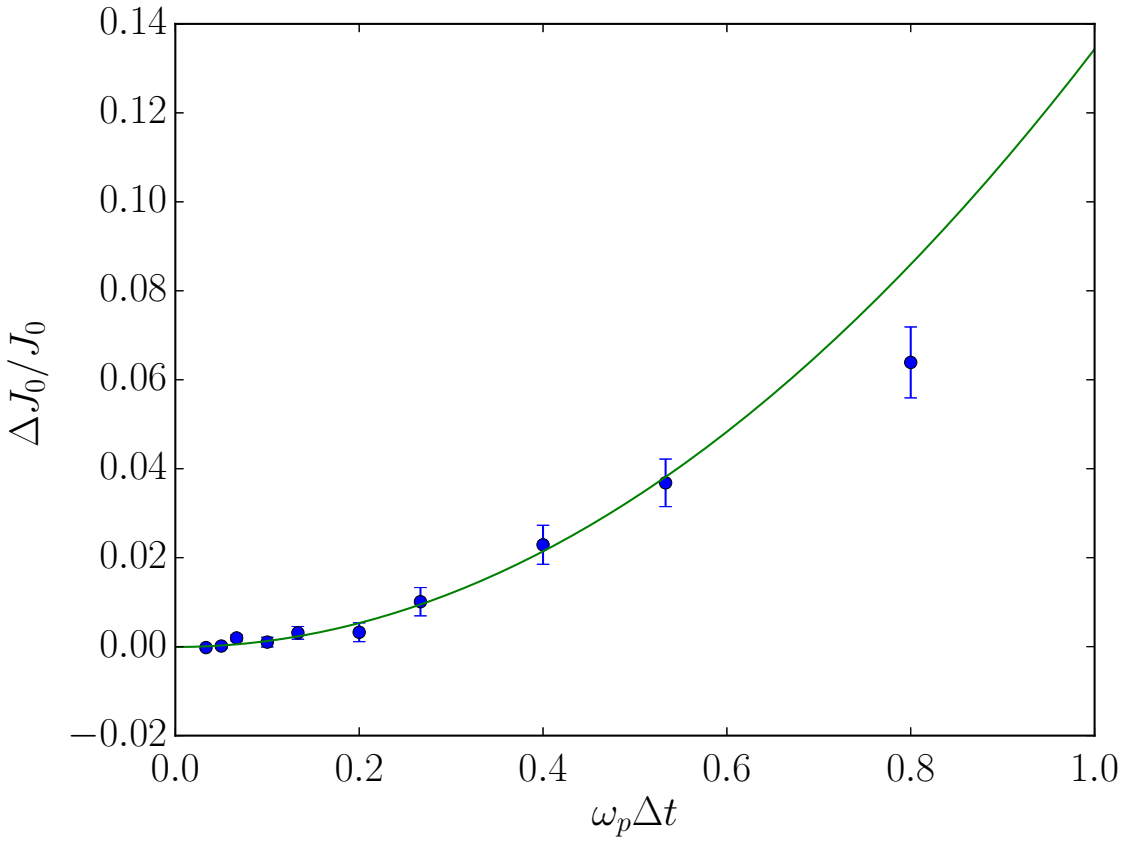

Figure 8. The difference between the current density observed in simulation and the corresponding exact solution, ΔJ_0, normalized by the exact result J_0. Expected behaviour is

$$\lim_{\omega_p \Delta t \to 0} \Delta J_0 / J_0 = 0,$$

with a rate of convergence $\propto (\omega_p \Delta t)^2$. Simulation data are shown by points with statistical error bars, and the solid curve is a least squares fit, from which the limiting value is estimated as $(-0.8 \pm 2.6) \times 10^{-4}$. The asymptotic rate of convergence is not reached for $\omega_p \Delta t = 0.8$.

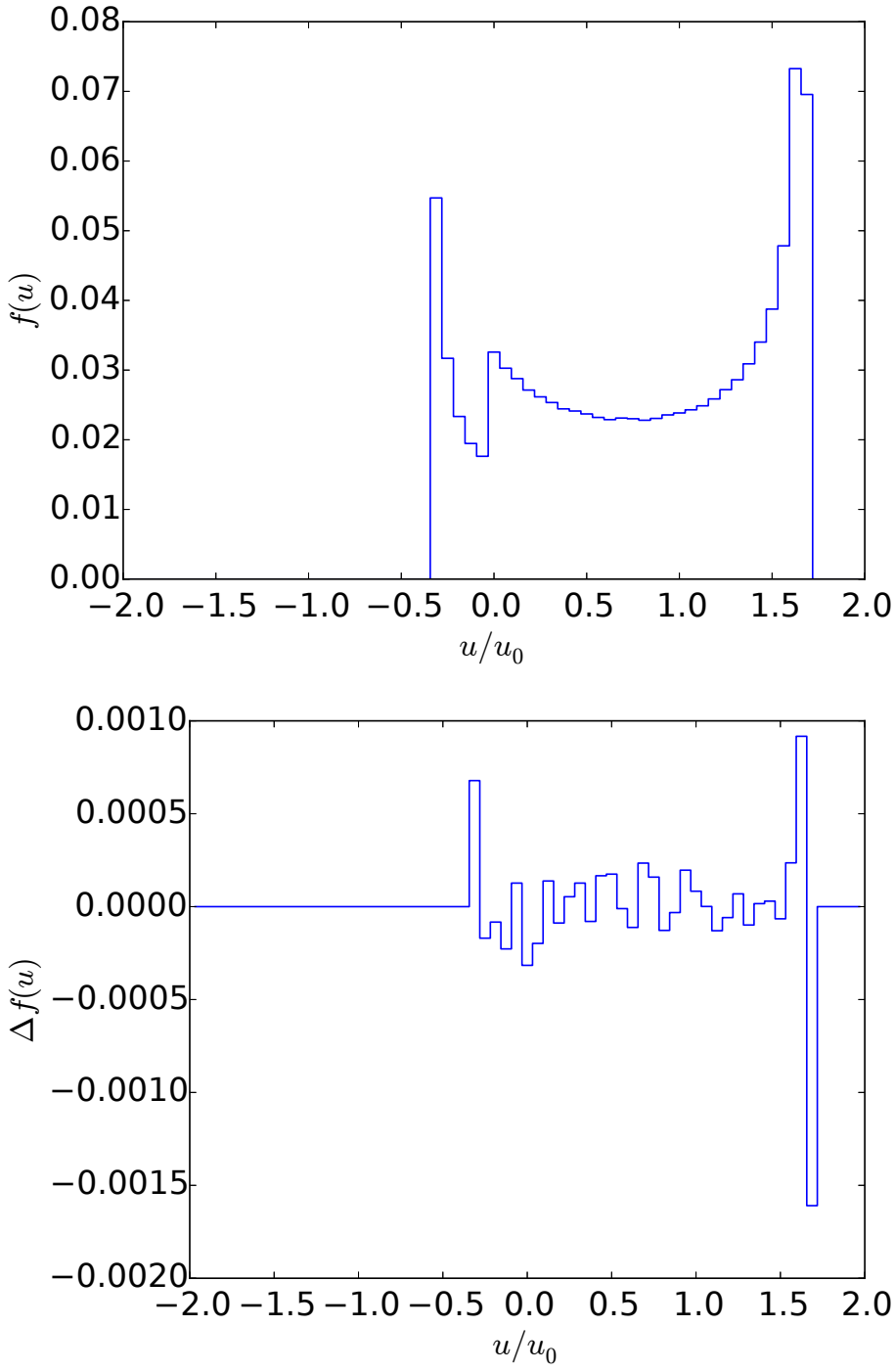

Figure 9. ~~An~~ The upper panel shows an example of the ion velocity distribution ~~determined~~ computed by the ~~theory of Kumar [14] , compared with the result of a simulation using the~~ present code. The ~~difference is~~ lower panel shows the differences between these data and the exact theory of Kumar [14] . These differences are evidently small, but by inspection, ~~is~~ are not obviously of a purely statistical character.

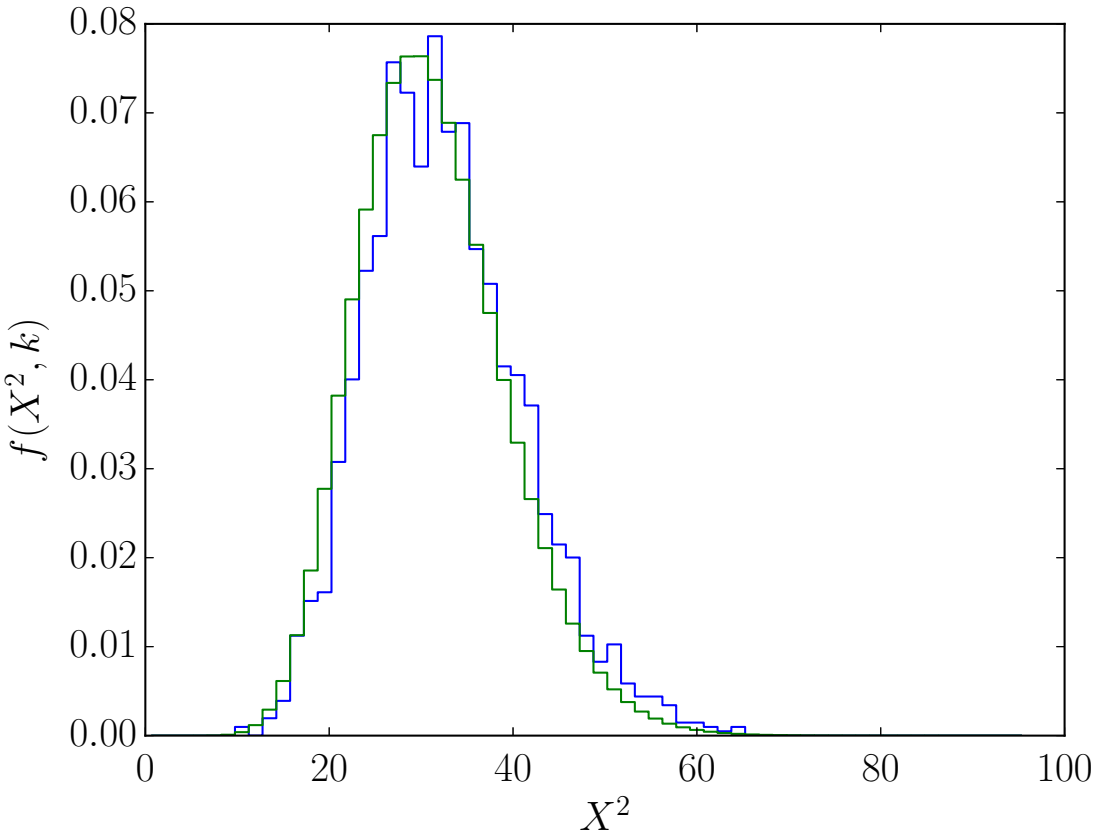

Figure 10. The distribution of X^2 values computed using eq. 16 for an ensemble of 2048 statistically independent ion velocity distribution functions of the kind shown in fig. 9. This distribution is compared with the expected distribution given by eq. 17.